LA-UR-04-3422



*Title:* MSTK: A Flexible Infrastructure Library for Developing Mesh-based Applications

*Author(s):* Rao V. Garimella

*Submitted to:* Preprint of article submitted to the 13th International Meshing Roundtable, Williamsburg, VA, Sep 19-22, 2004.

# MSTK - A FLEXIBLE INFRASTRUCTURE LIBRARY FOR DEVELOPING MESH BASED APPLICATIONS

Rao V. Garimella

*Los Alamos National Laboratory, Los Alamos, NM, USA, rao@lanl.gov*

## ABSTRACT

MSTK is a powerful framework for low-level creation and manipulation of unstructured meshes. MSTK is not a mesh generator but it can be used to develop advanced mesh generation software and other mesh-based applications. The salient feature of MSTK is that it is allows multiple mesh representations while presenting a common functional interface to the developer. This allows application developers to use a mesh representation optimized for their particular algorithms. MSTK allows developers to focus on their applications rather than on the details of mesh data structures.

**Keywords: meshing, data structures**

## 1. INTRODUCTION

A survey of several review web sites[1,2] for mesh generation software shows that there are over 100 monolithic programs designed to help users generate unstructured meshes [1, 2, 3, 4]. Combined with other applications for numerical analysis, mesh adaptation and graphical visualization, the amount of software to generate and use unstructured meshes is huge.

All these applications, undoubtedly, have a common set of needs for representing and manipulating unstructured meshes. The typical infrastructure needed by these applications is a set of data structures for representing the mesh and software mechanisms for accessing and modifying mesh data. A set of data structures representing the mesh or a *mesh representation*, consists of topological *mesh entities* (vertices, edges, faces, regions) and *topoogical adjacencies* (inter-entity connections, e.g. face-edge, edge-region, region-vertex). Mesh representations differ in the specific entities and adjacencies they choose to explicitly store in their data structures. The combination of a mesh representation and a set of access mechanisms for mesh data is called a *mesh framework* or *meshing infrastructure*.

Although the need for common infrastructure to enable the rapid and efficient development of mesh based programs is obvious, efforts to address this issue have not gotten underway until very recently [5, 6].

One reason for the scarcity of general meshing infrastructure is that the needs of the various meshing and analysis applications vary widely and there is little agreement on the computational efficiency of different mesh data representation. Therefore, a large number of mesh representations are in use in the computational community, each tailored to a specific application. Some simple numerical analysis programs use only a minimal representation consisting of elements (quads, tetrahedra, etc.) defined by nodes (or points). Other, complex applications find it necessary augment the basic element-node data structures with additional data like element-to-element or node-to-node connectivity. Even more sophisticated applications, like meshers working directly from CAD models, evolving geometry analysis procedures, etc., find it useful to use a much richer mesh representation consisting of a full set of mesh entities with a extensive set of topological adjacencies [7, 8]. Therefore, to

[1] `http://www.andrew.cmu.edu/user/sowen/softsurv.html`

[2] `http://www-users.informatik.rwth-aachen.de/~roberts/software.html`

gain widespread acceptance, it is important to have a flexible mesh framework which allows applications to choose a mesh representation that closely matches their needs [5, 8]. At the same time, the infrastructure should be lightweight and efficient to have sufficient utility for real world applications.

Fortunately, the need for general meshing infrastructure is increasingly being recognized and several development efforts have been introduced in the last few years. These include the Algorithm Oriented Mesh Database[3] (AOMD) [5] and the Sandia Mesh Database (MDB) Component[4] [6]. Both AOMD and MDB are flexible representation mesh frameworks but AOMD allows flexibility in the types of mesh entities and types of topological adjacencies stored while MDB allows flexibility only in the types of adjacencies stored. Both AOMD and MDB are implemented using C++ in order to utilize the object-oriented programming constructs offered by the language. In addition, AOMD depends on a particular implementation of the Standard Template Library (STLport) for providing constructs such as container classes. There are a few other mesh representation frameworks such as OpenMesh[5], GrAL[6] (Grid Algorithms Library), Libmesh[7] and GTS[8] (GNU Triangulation Software) but these are much less general than AOMD or MDB, restricting themselves to special types of meshes such as triangular or polygonal surface meshes.

MSTK (**M**e**S**h **T**ool**K**it) is a newly developed, flexible mesh framework that provides mesh-based application developers low-level infrastructure for reading, writing, creating and manipulating unstructured meshes without having to design and implement their own mesh data structures. MSTK is designed to be powerful, lightweight and efficient making it suitable for both rapid development of algorithms as well as incorporation into large, real world applications. For this reason, MSTK utilizes many object-oriented programming principles but is implemented in C to provide maximum efficiency. Also, while MSTK is designed to allow general combinations of entities and adjacencies, it also encodes certain commonly used representations in order to increase its efficiency when using these mesh representations. Some of these representations encode adjacency types that are not supported in other general frameworks, such as element-to-element connectivity.

In this paper, a brief description of MSTK is provided including details on the design of the framework and the software mechanisms for interacting with it. MSTK is in development and some of the capabilities described here are still in initial development phase. However, preliminary versions of the software have successfully been used to implement several applications such as mesh smoothing, 2D mesh reconnection, non-conforming refinement of polygonal meshes (also known as Adaptive Mesh Refinement or AMR) and 3D remapping within Los Alamos National Laboratory. MSTK is currently available to Los Alamos National Laboratory researchers but efforts are underway to make MSTK available for more widespread use.

[3] http://www.scorec.rpi.edu/AOMD
[4] htpp://sass1693.sandia.gov/cubit/mdb.htm
[5] http://www.openmesh.org
[6] http://www.math.tu-cottbus.de/~berti/gral/
[7] htpp://libmesh.sourceforge.net
[8] htpp://gts.sourceforge.net

## 2. MESH REPRESENTATIONS IN MSTK

The success of general mesh frameworks depends on their ability to satisfy the requirements of a wide variety of applications such as visualization, mesh generation and adaptation, and numerical analysis applications. It is impossible for these frameworks to optimally serve the needs of all these applications with a single method for representing the mesh. Therefore, it is becoming increasingly common for generic mesh frameworks to provide application developers the ability to customize the mesh representation to suit their algorithms.

Two such flexible mesh frameworks are AOMD [5] and Sandia MDB [6]. Both AOMD and MDB provide users the ability to prescribe the set of adjacencies required in the mesh representation. In addition, AOMD provides the ability to prescribe the entities that will be explicitly stored in the representation. However, it appears unavoidable that providing users the ability to choose an arbitrary set of topological entities and adjacencies at run-time will carry a high computational cost. This is because such a capability will require to software to repeatedly check if an entity or an adjacency is readily available or must be computed before answering any query.

MSTK approaches the problem of providing flexibility in mesh representations to application developers differently. It defines a set of predefined mesh representations that are commonly used and has specific code for adjacency retrieval based on the particular representation. Switching to the appropriate code is done at runtime using jump tables. Since the code being invoked is tailor-made for that representation, the code execution is expected to be faster than checking for the general case. Still, in order to accomodate the varied needs of applications, MSTK will eventually incorporate the ability to have a general, dynamically defined mesh representation, albeit at a higher cost.

The representations that are incorporated into MSTK are taken from a comparative study of the efficiency of various mesh data structures [8]. Of the ten repre-

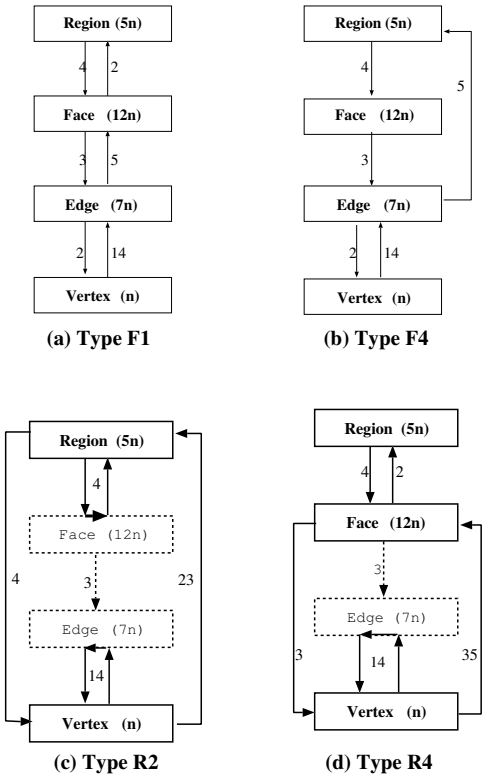



**Figure 1**: Mesh representations of MSTK

sentations, discussed in [8], four representations (Figure 1) have been incorporated into the code. Of these, representations F1 and F4 are full representations, i.e., entities of all dimensions (up to the dimension of the mesh) are explicitly represented, and have undergone extensive testing. Representation R2 is a reduced representation that only represents elements (faces or regions) and nodes. The adjacency set for representation R2 includes element-to-element and node-to-node connections. Representation R4 is a reduced representation in which edges are never represented explicitly and node-to-node connections are included. The inclusion of representations such as R2 and R4 set MSTK apart from other general mesh frameworks that insist on containing only inter-dimensional adjacencies. Both R2 and R4 are still being developed and tested. MSTK can also be modified to include other types of mesh representations, should they be needed.

## 3. MESHES, MESH ENTITIES AND OTHER CONSTRUCTS

Unstructured meshes are natural candidates for application of the object-oriented paradigm and therefore, MSTK also views a mesh and its components as "objects" although the code is implemented in C for efficiency. The framework emulates ideas such as data hiding, class hierarchy, private and public methods as far as possible.

The primary construct in MSTK is the mesh and MSTK can support multiple meshes simultaneously. The mesh contains references to the entities it is made up of, the geometric model it refers to (if available) and the attributes defined on it.

The types of mesh entities in MSTK are the usual vertices, edges, faces and regions corresponding to topological entities of dimension 0, 1, 2, and 3. At this time, only vertices have any geometry associated with them in terms of their coordinates. The capability to have general curved geometry associated with edges and faces will be incorporated in the future. In terms of topology, MSTK supports straight-sided polygonal faces and polyhedral regions.

Sets of entities in MSTK are handled as dynamic lists, which can be created, destroyed, modified or queried using multiple mechanisms.

MSTK also provides the ability to define mesh attributes which can be associated with some or all entities of the mesh. Attribute values can be of type integer, double precision real or a pointer. The pointer type attribute allows storage of more complex data types such as tensors as well as handles to other entities (e.g., parent-child relationships or relationship of entities in two meshes).

All MSTK constructs are referenced by handles to avoid direct access of internal data by applications.

## 4. CLASSIFICATION OR MESH-MODEL RELATIONSHIPS

The concept of a formal mesh-model relationships or *classification* was first introduced in early mesh generation papers by Shephard et.al. [9, 10]. *A mesh entity is said to be **classified** on a geometric model entity of the same or higher topological dimension if the mesh entity forms all or part of the discretization of the geometric model entity.* A mesh entity cannot be classified on a model entity of lower dimension. The concept of classification is very useful for meshing applications since it allows them to verify that the mesh conforms to the topology of the geometric model or violates it in known ways. Classification is also useful for mesh-based numerical analysis applications since it simplifies specification of analysis attributes such as boundary conditions and material properties.

However, the reality of mesh generation and numerical analysis applications is that not all meshes come from a strictly defined geometric model. Sometimes a mesh is defined with respect to a simple abstract model, e.g.

a complex mesh for a Rayleigh-Taylor simulation is defined on a simple rectangular domain. Still others may have no definition of an underlying geometric domain at all as is the case for meshes defined from range data or 3D scanners.

To support these varying applications, MSTK supports classification of mesh entities at multiple levels of detail. If a geometric model is available, a geometric model entity can be associated with a mesh entity. If not, only the dimension or the dimension and ID of an abstract geometric model entity can be associated with a mesh entity. Finally, a mesh entity can have no geometric model information associated with it if such knowledge is unavailable.

This flexibility in specifying classification information allows application to use this powerful concept while not forcing them to always work with a formal geometric modeling package.

## 5. MSTK FUNCTIONAL INTERFACE

MSTK provides an extensive functional interface to enable applications to create, modify and query meshes and mesh entities in a variety of ways. All MSTK "objects" including meshes, mesh entities, lists and attributes have "constructors","destructors", operators to modify their data and operators to query their data. All mesh entities contain upward, downward and same-level adjacency information depending on the representation being used. Regardless of which information is really stored, all adjacency queries are possible for mesh entities.

The deletion of mesh entities in MSTK takes two forms. The first form of the deletion operation destroys the entity and all its associated data. It also removes the entity from the mesh and breaks all upward and downward adjacency relationships involving it. The second form of the deletion operation only makes the entity invisible to the mesh and other entities that refer to it through adjacency relationships. The entity itself and its data are not destroyed. This allows applications to modify a mesh but also keep the original mesh information around.

A special set of operators available in MSTK are operators for marking mesh entities. Markers are similar to integer attributes that can only take the value of 0 or 1; however, they are implemented much more efficiently than attributes. Marking of entities has enormous utility in mesh algorithms involving the union of entity sets, often reducing algorithm complexity from $O(n^2)$ to $O(n)$ [8].

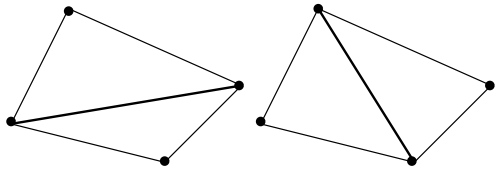


**Figure 2**: Edge Swap in 2D

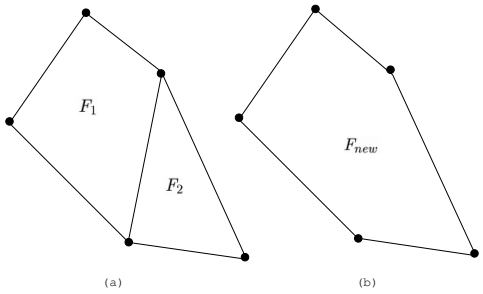



**Figure 3**: Joining two faces along a common edge

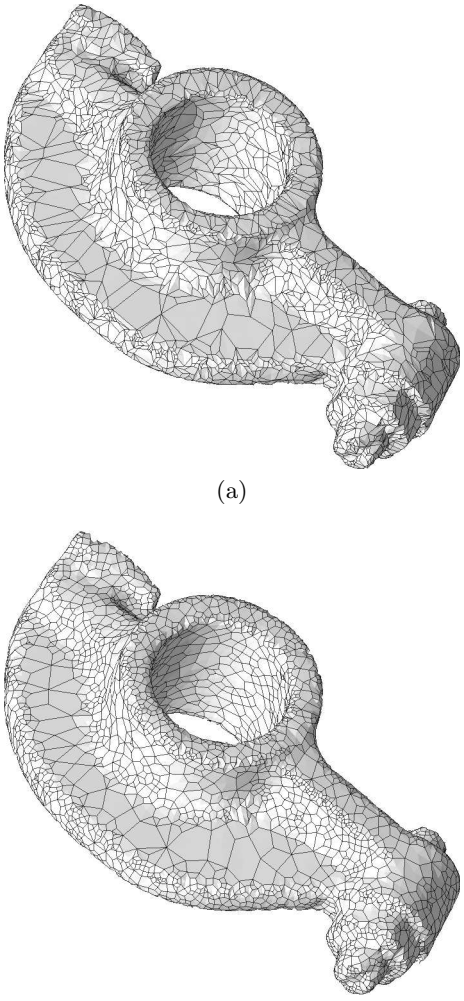
## 6. ADDITIONAL FUNCTIONALITY IN MSTK

MSTK also aims to provide frequently used higher level mesh functionality to applications. These functions perform operations such as swapping an edge in 2D (Figure 2), joining two faces along a common edge (Figure 3), evaluating the condition number quality measure of a face, etc. These functions are being incorporated into MSTK as and when they are being developed for other applications.

## 7. APPLICATION EXAMPLES

MSTK has been incorporated into a number of in-house applications for meshing and analysis procedures. It has been used to implement mesh improvement strategies for surface and volume meshes [11, 12] (see Figure 4). It has also been used for locally conservative remapping of solution data for general polyhedral grids [13] and for a mesh adaptation method using mesh reconnection. Finally, MSTK has been used to develop an non-conforming adaptive mesh refinement (AMR) procedure for adaptive simulations using mimetic discretization of elliptic and parabolic PDEs on polygonal meshes.

The example below illustrates the AMR capability on a simple diffusion problem. Consider a square domain with a high diffusion coefficient in the lower-left and upper-right quadrants and a low diffusion coefficient elsewhere. Neumann boundary conditions are imposed



(a)

(b)

**Figure 4**: Initial and Optimized meshes of a rocker arm obtained from an optimization procedure developed using MSTK

on the top and bottom boundaries. The pressure is set to 1.0 at the left boundary and 0 at the right. The intensity isolines and streamlines for the problem are shown in Figures 5a,b respectively. The adaptively refined mesh required to reduce the error to less than $10^{-3}$ is shown in Figure 5c. As seen in the figure, the maximum refinement occurs where the problem has a singularity.

## 8. FUTURE WORK

MSTK is constantly being developed to make it faster, more compact and more functional. Many new capabilities are being planned for MSTK in the near future, some of which are described below.

As mentioned earlier, only two full mesh representations are implemented and tested in MSTK. The addition of more mesh representations will make MSTK more appealing to a wider range of application developers. In particular, the deployment of reduced representations will be of enormous benefit to applications that are willing to pay a higher computational cost in order to accommodate larger meshes. Similarly, the capability to have user defined set of adjacencies in MSTK will also have considerable utility.

While MSTK is reasonably memory efficient, much work is expected to be done soon for making it more compact using advanced data encoding techniques. Initial estimates indicate that the number of elements that MSTK can represent will at least be tripled by these improvements.

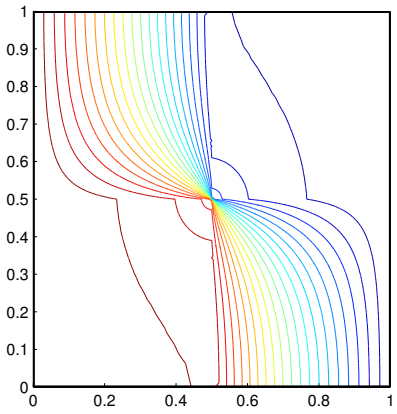
Also, a major effort for making MSTK capable of handling distributed meshes for parallel computing is expected to be undertaken in the near future.

Finally, higher level mesh manipulation and query functions continue to be added to MSTK as development continues on other applications using MSTK.

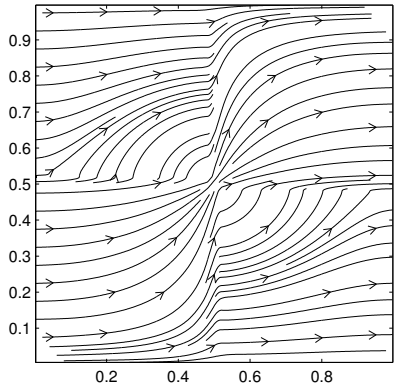
## 9. CONCLUSIONS

A software infrastructure, called MSTK, for low level manipulation of unstructured meshes in a general framework was described. MSTK is capable of representing unstructured meshes in multiple ways while presenting a uniform, easy-to-use interface to application developers. MSTK offers application developers a powerful, yet lightweight and efficient infrastructure for managing unstructured meshes. This allows developers to focus on their primary high-level goals instead of being bogged down in mesh data structure coding. MSTK is already being used for a number of meshing and analysis applications, and is quite stable.

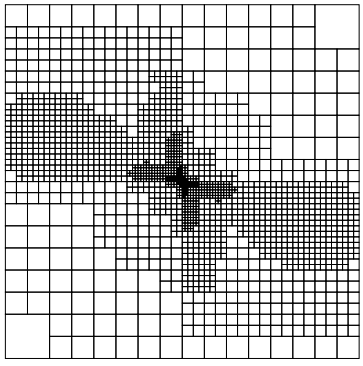
Future work on MSTK include making MSTK capable of handling distributed meshes, boosting its storage




(a)




(b)



(c)

**Figure 5**: Simulation of diffusion problem with relatively high coefficients in lower-left and upper-right quadrants (a) Isolines of pressure (b) Streamlines (c) AMR mesh

efficiency, adding new representations and developing more high level mesh manipulation functionality.

More information on MSTK is available from `http://math.lanl.gov/~rao/Meshing-Projects/MSTK`.

## 10. ACKNOWLEDGMENTS

The work of Rao V. Garimella was performed at Los Alamos National Laboratory operated by the University of California for the US Department of Energy under contract W-7405-ENG-36. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

The author thanks Dr. K. Lipnikov for the use of simulation results shown in Figure 5.

## References

[1] Joe B. "GEOMPACK - A Software Package for the Generation of Meshes Using Geometric Algorithms." *Advances in Engineering Software*, vol. 56, no. 13, 325–331, 1991

[2] "TetMesh - GHS3D, Ver. 3.1." Tech. rep., INRIA/SIMULOG, Guyancourt, France, 2001. (`http://www.simulog.fr/tetmesh/`)

[3] "LaGriT - Los Alamos Grid Toolbox." Tech. rep., Los Alamos National Laboratory, Los Alamos, NM, 1995. (`http://www.t12.lanl.gov/home/lagrit/`)

[4] The CUBIT Group. "CUBIT Mesh Generation Toolkit." Tech. rep., Sandia National Laboratories, Albuquerque, NM, 2001. (`http://sass1693.sandia.gov/cubit`)

[5] Remacle J.F., Karamete B.K., Shephard M.S. "Algorithm Oriented Mesh Database." *Proceedings of the Ninth International Meshing Roundtable*, pp. 349–359. Sandia National Laboratories, New Orleans, LA, Oct 2000. Sandia Report SAND 2000-2207

[6] Tautges T.J., Merkley K., Stimpson C.J., Meyers R.J. "The Sandia Mesh Database Component (MDB)." *Proceedings of the Seventh Us National Congress on Computational Mechanics*. Albuquerque, NM, Jul 2003. (`http://sass1693.sandia.gov/cubit/mdb.htm`)

[7] Beall M.W., Shephard M.S. "A General Topology-Based Mesh Data Structure." *International Journal for Numerical Methods in Engineering*, vol. 40, no. 9, 1573–1596, May 1997

[8] Garimella R.V. "Mesh Data Structure Selection for Mesh Generation and FEA Applications." *International Journal of Numerical Methods in Engineering*, vol. 55, no. 4, 451–478, Oct 2002

[9] Shephard M.S., Finnigan P.M. "Integration of geometric modeling and advanced finite element preprocessing." *Finite Elements in Analysis and Design*, vol. 4, no. 2, 147–162, Aug 1988

[10] Shephard M.S., Georges M.K. "Automatic Three-Dimensional Mesh Generation by the Finite Octree Technique." *International Journal for Numerical Methods in Engineering*, vol. 32, no. 4, 709–749, 1991

[11] Garimella R.V., Shashkov M.J., Knupp P.M. "Triangular and Quadrilateral Surface Mesh Quality Optimization using Local Parametrization." *Computer Methods in Applied Mechanics and Engineering*, vol. 193, no. 9-11, 913–928, Mar 2004

[12] Dyadechko V. *Geometrically Adapted Meshes and Iterative Solvers for Elliptic Problems*. Ph.D. thesis, University of Houston, May 2003

[13] Garimella R., Kucharik M., Shashkov M. "Efficient Algorithm for Local-bound-preserving Remapping in ALE methods." *Proceedings of The European Conference on Numerical Mathematics and Advanced Applications (ENUMATH 2003)*. Prague Czech republic, Aug 2003